Artificial and Computational Intelligence

AIMLCZG557

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Presented by
V Indumathi –Guest Faculty –BITS-WILP
indumathi.p@wilp.bits-pilani.ac.in

innovate achieve lead

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

AIMLCZG557 – CS#4

Problem Formulation & Uninformed Search Algorithms

ACI: Disclaimer and Acknowledgements

⚠

- Few content for these slides may have been obtained from prescribed books and various other source(s) on the Internet.
- We hereby acknowledge all the contributors for their material and inputs and gratefully acknowledge people others who made their course materials freely available online. We have provided source information wherever necessary.
- We have added and modified the content to suit the requirements of the class dynamics & live session's lecture delivery flow for presentation.
- This *is not a full-fledged* reading materials. Students are requested to refer to the textbook w.r.t detailed content as per the course handout as shared over e-learning portal - Taxila.
- **Slide Source / Preparation / Review:**
- From BITS Pilani WILP**:** Prof. Rajavadhana, Prof. Indumathi, Prof. Sangeetha and Prof Parthasarathy PD
- From BITS On-campus & External : Mr.Santosh GSK

Problem formulation

↳ Goal / Obj < min / max } PS

↳ PS → IS, Actions, Tr. f, P.cost, Goal

PS (Basic info)

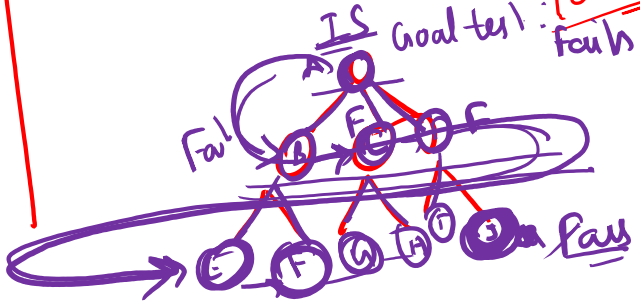
Uninformed
Blind search

↳ BFS – Queue
↳ DFS
↳ IDS
↳ UCS

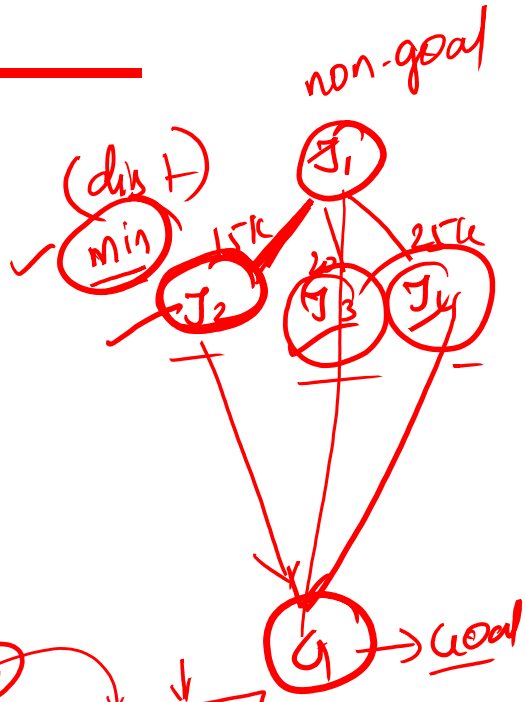
Informed
Basic + addn. info
h(n) → Domain

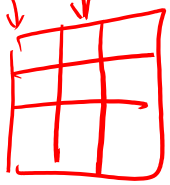
↳ BFS
A*

A1 (30 min) | A2 (60 min)



50
Goal J

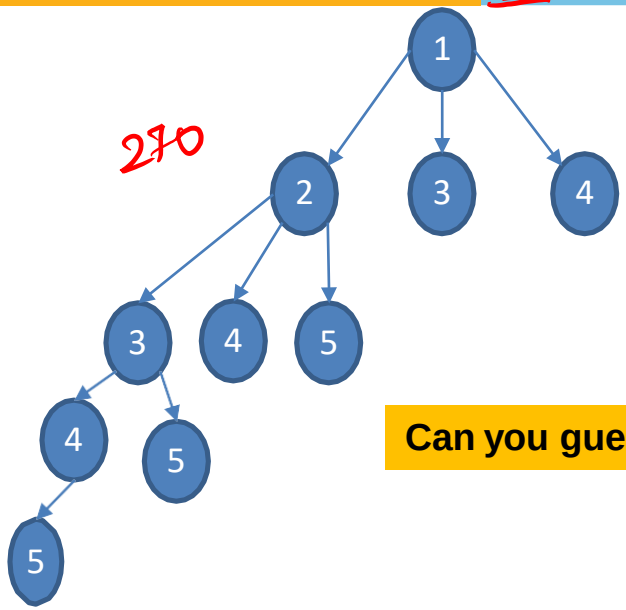




BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

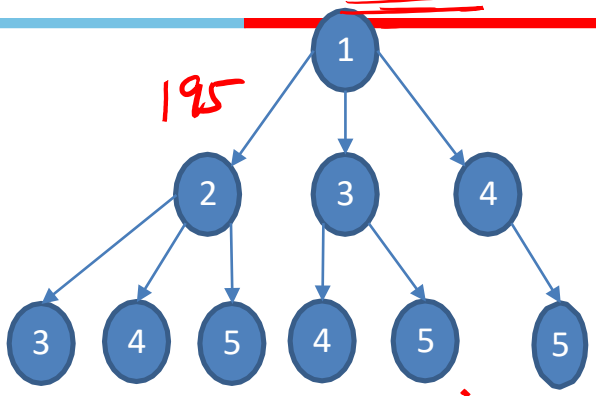
Search Algorithm – Uninformed Example

DFS

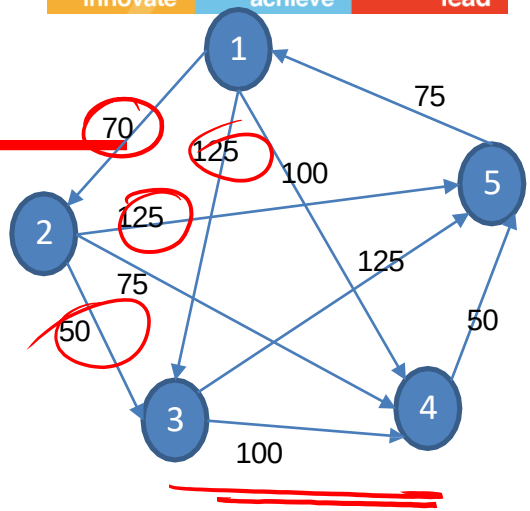


270

BFS



195



I/P

Can you guess which algorithm are these ?

1-2-5

1-2-3-4-5

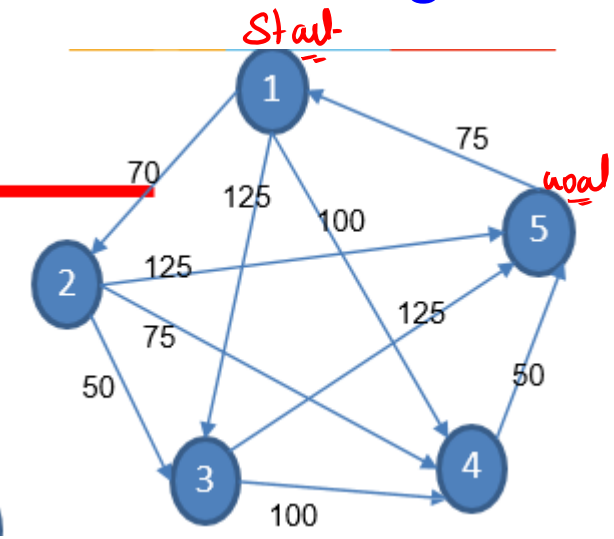
1-3-5

UCS

1-4-5 ⇒

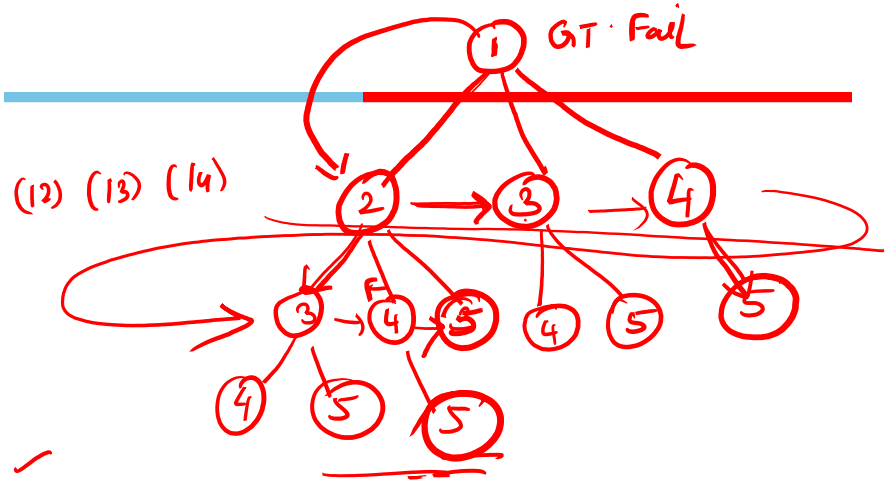
100 + 50 ⇒ 150

Search Algorithm – Uninformed Example



BFS

ST



openlist	closed list	goal
(1)	–	Fail (1)
(12)(13)(14)	(1)	Fail((12)
(13)(14)(123)(124)(125)	(1)(12)	Fail (13)
(14)(125)(124)(125)(134)(135)	(1)(12)(13)	Fail (14)
(123)(124)(125)(134)(135)(145)	(1)(12)(13)(14)	Fail (123
(124)(125)(134)(135)(145)(1234)(1235)	" (123)	Fail (124)
(125)(134)(135)(145)(1234)(1235)(1245)	" (124)	PASS (125)

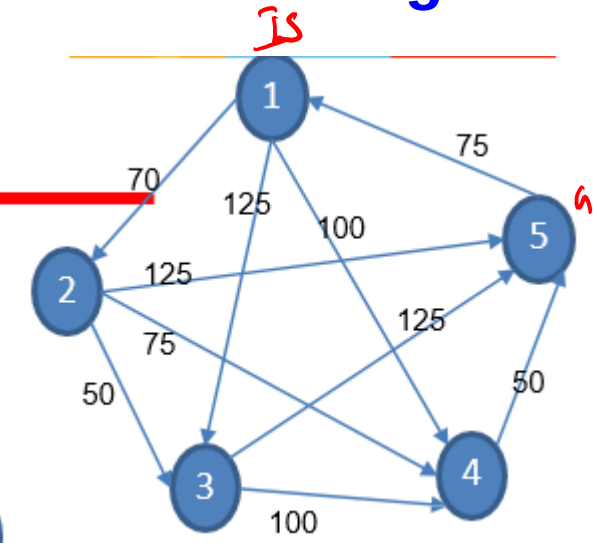
PATH - 1-2-5

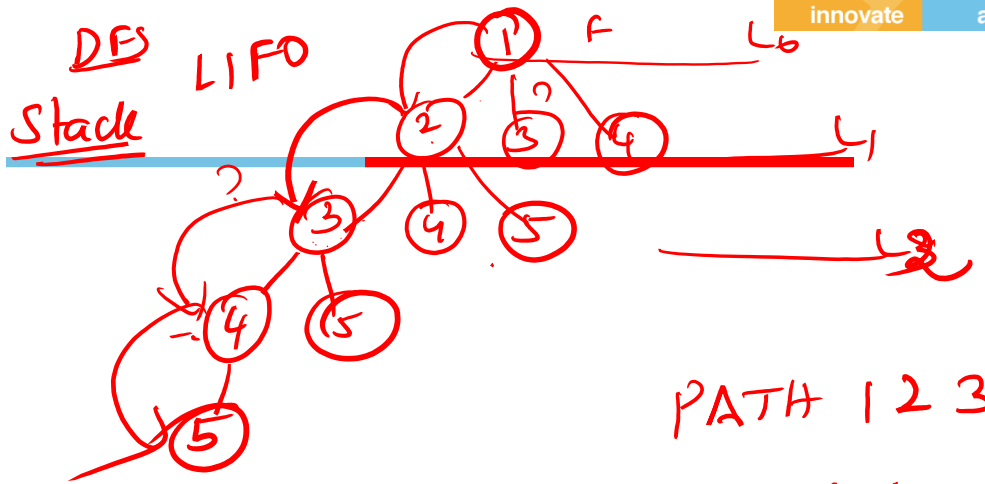
" COST 70 + 125

⇒ 195

BFS

Search Algorithm – Uninformed Example





PATH 1 2 3 4 5

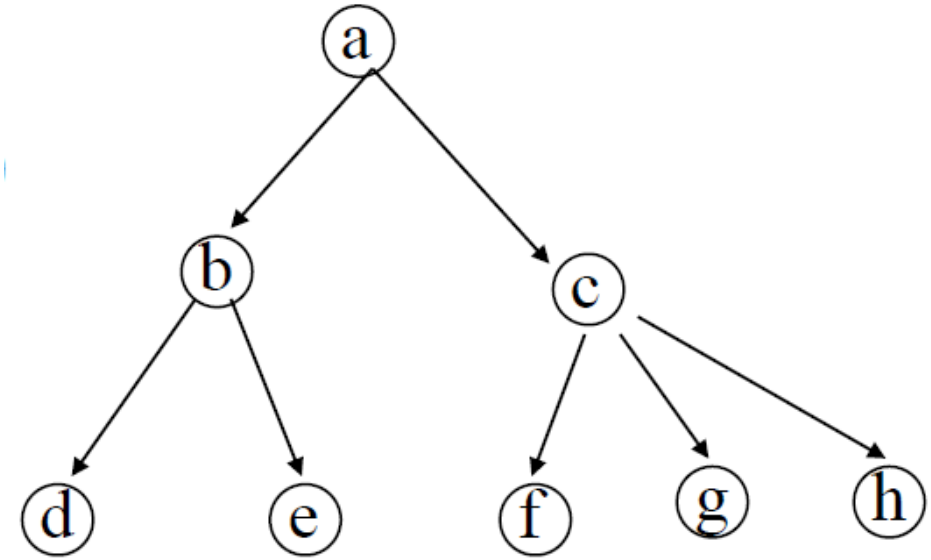
70 + 50 + 100 + 50

=> 270

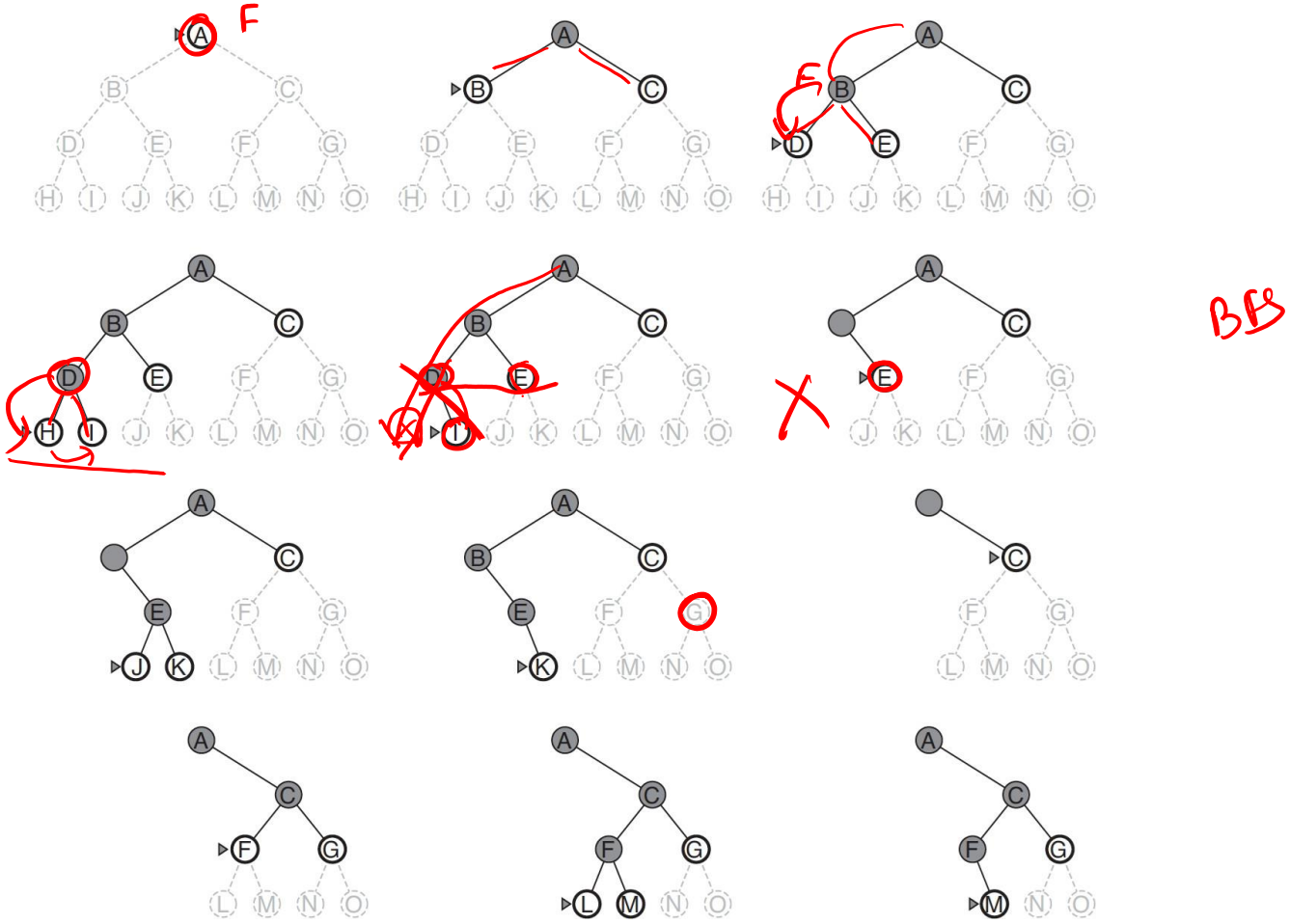
open list	closed	Goal
(1)	–	F(1)
(12) (13) (14)	(1)	F(12)
(123) (124) (125) (13) (14)	(1) (12)	F(123)
(1234) (1235) (124) (125) (13) (14)	(1) (12) (123)	F(1231)
(12345) (1235) (124) " "	" (1234)	Path (12345)

Depth First Search (DFS) Summary

- Depth-first visit order is "*aggressive*" in the sense that it examines complete paths
- Maintain *stack* of nodes to visit.



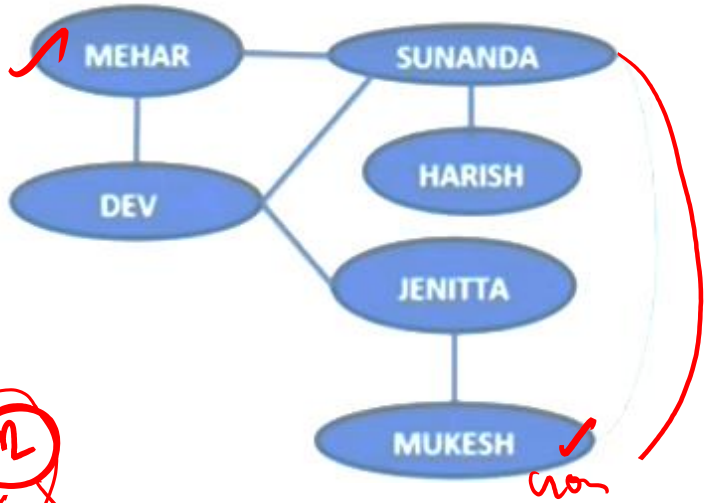
Depth First Search (DFS) Summary



Applications

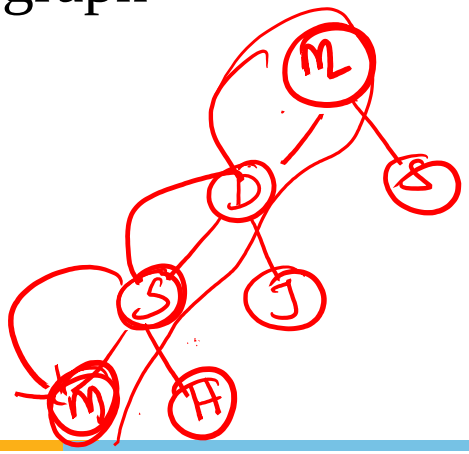
Breadth First Search

- Finding path in a graph (many solutions)
- Finding the Bipartitions in a graph



Depth First Search

- Find the Connectedness in a graph
- Topological Sorting



Depth First Search (DFS)

Completeness – Complete in finite state spaces because it will eventually expand every node

Optimal – Not Optimal as it would stop when the goal node is reached without evaluating if there is a better path

Time Complexity - $\mathcal{O}(b^m)$ where m = maximum depth of any node

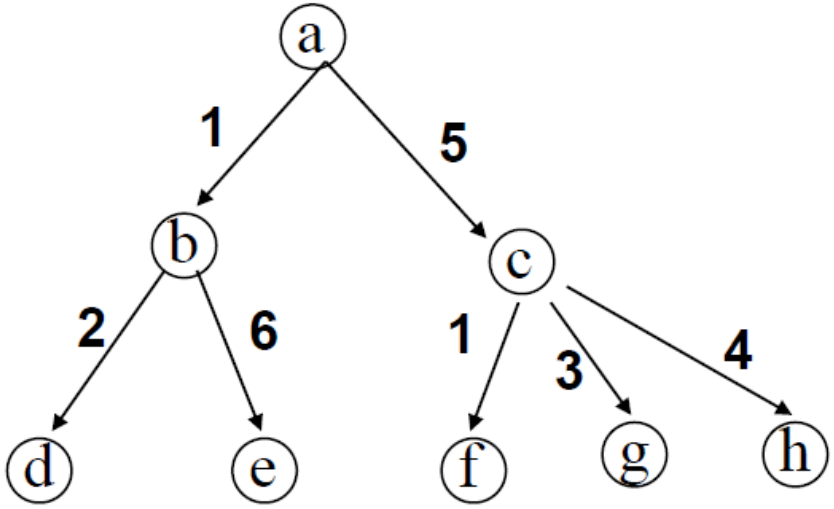
- Can be much larger than the size of state space
- m can be much larger than d (shallowest goal)

Space Complexity – Needs to store only one path and unexpanded siblings.

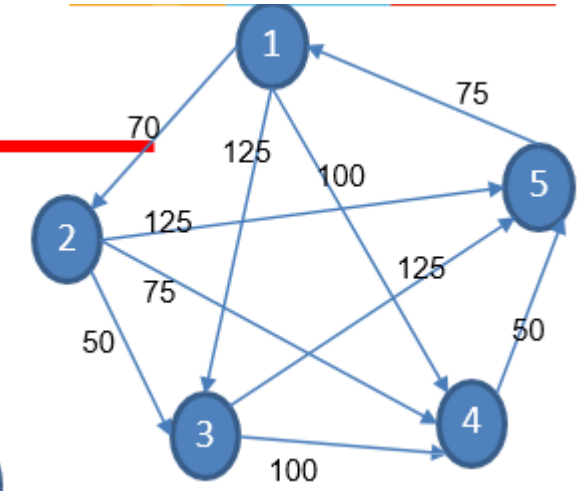
- Any node expanded with all its children can be removed from memory
- Requires storage of only $\mathcal{O}(bm)$, b – branching factor, m - max depth

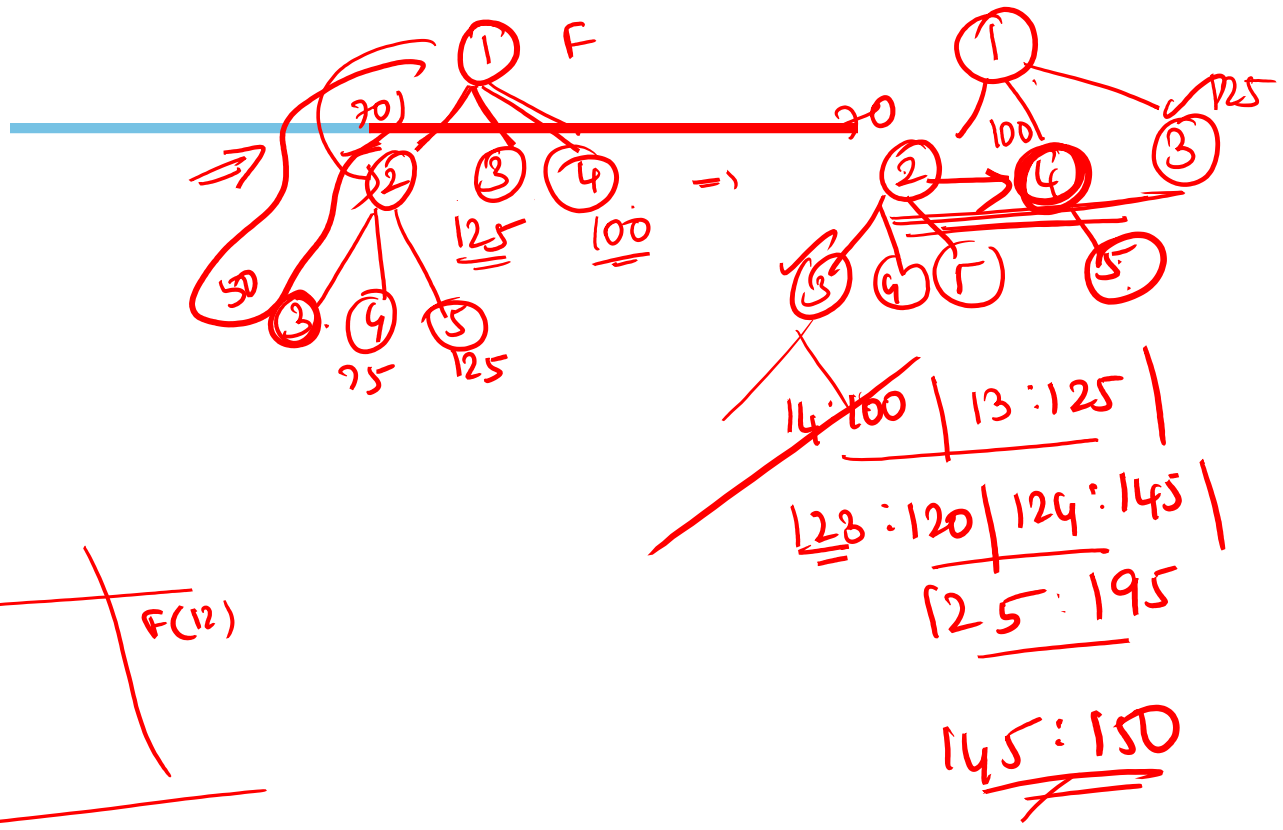
Uniform Cost Search: Cheapest First

- Instead of expanding the shallowest node, Uniform-Cost search expands the node n with the lowest path cost g(n)
- **Sorting** the Frontier as a priority queue ordered by g(n)
- Goal test is applied during expansion
  - The goal node if generated may not be on the optimal path
  - Find a better path to a node on the Frontier

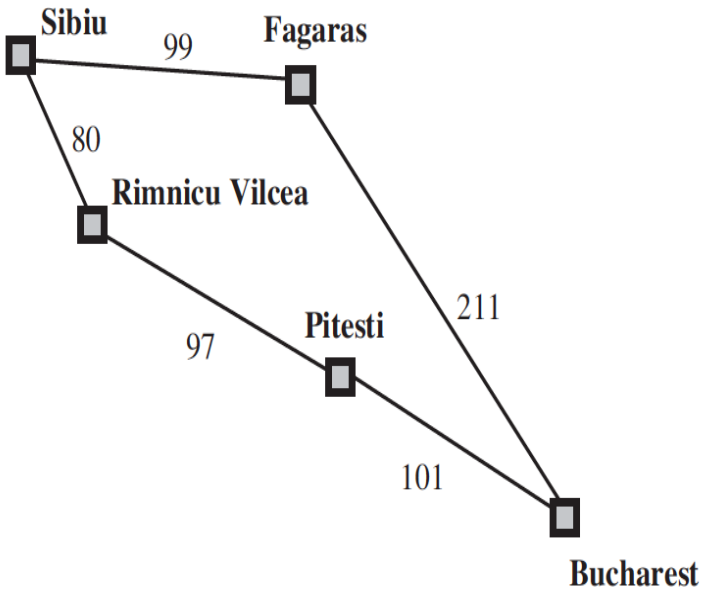


Search Algorithm – Uninformed Example





Uniform Cost Search: Cheapest First



- Current State: Sibiu
- Frontier: []
- Step: Expand Sibiu
- Result: Generates ("Rimnicu Vilcea" 80)

("Fagaras", 99)

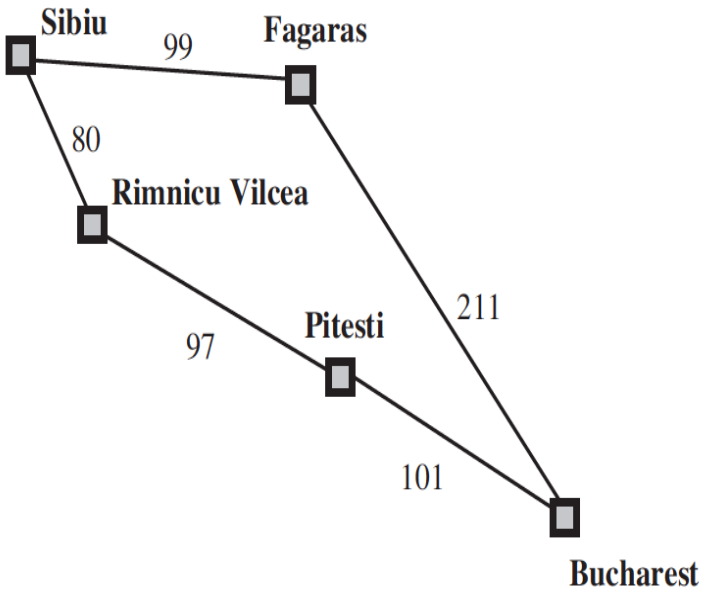
Add to Frontier

BFS

- Initial State: Sibiu
- Goal State: Bucharest

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

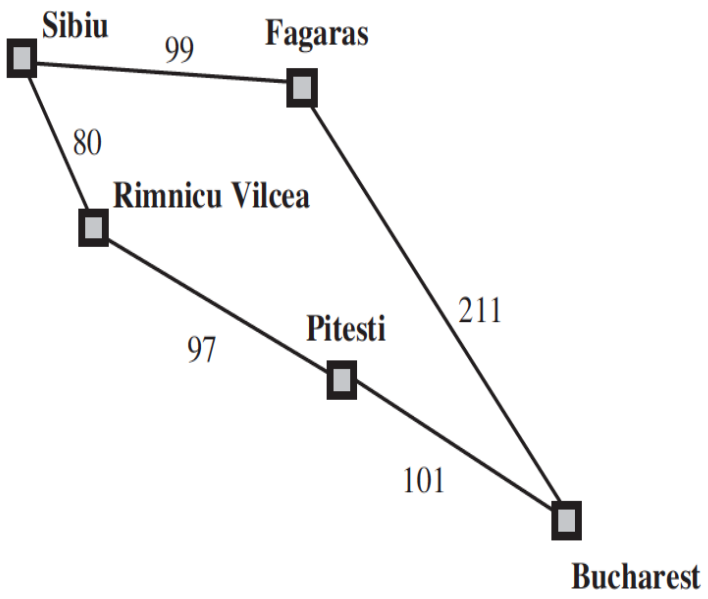
Uniform Cost Search: Cheapest First



- Current State: Sibiu
- Frontier: [("Rimnicu Vilcea" 80)
  ("Fagaras", 99)]
- Step: Expand "Rimnicu Vilcea"
  (least cost)
- Result: Generates ("Pitesti", 177)
  Add to Frontier

- Initial State: Sibiu
- Goal State: Bucharest

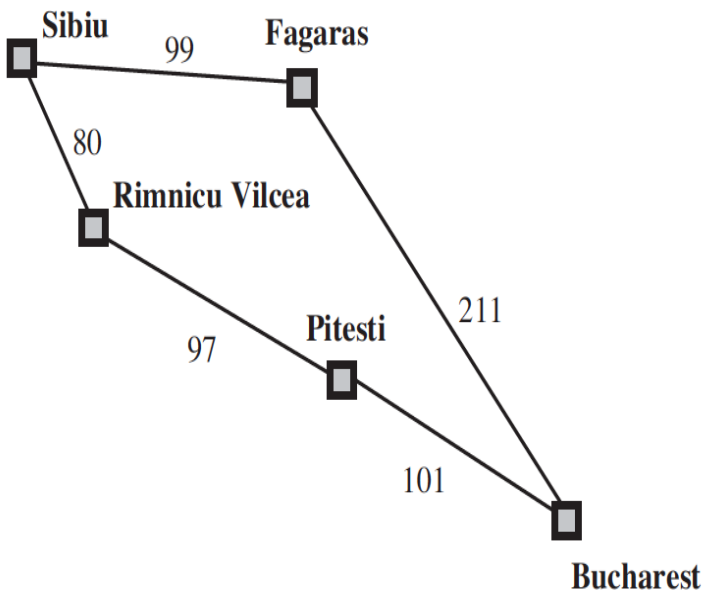
Uniform Cost Search: Cheapest First



- Initial State: Sibiu
- Goal State: Bucharest

- Current State: Rimnicu Vilcea (not a Goal state)
- Frontier: [("Fagaras", 99)
  ("Pitesti",177)]
- Step: Expand "Fagaras" (least cost)
- Result:
  - Generates ("Bucharest", 310)
  - Add to Frontier
  - (It's a Goal State but we won't test during generation)

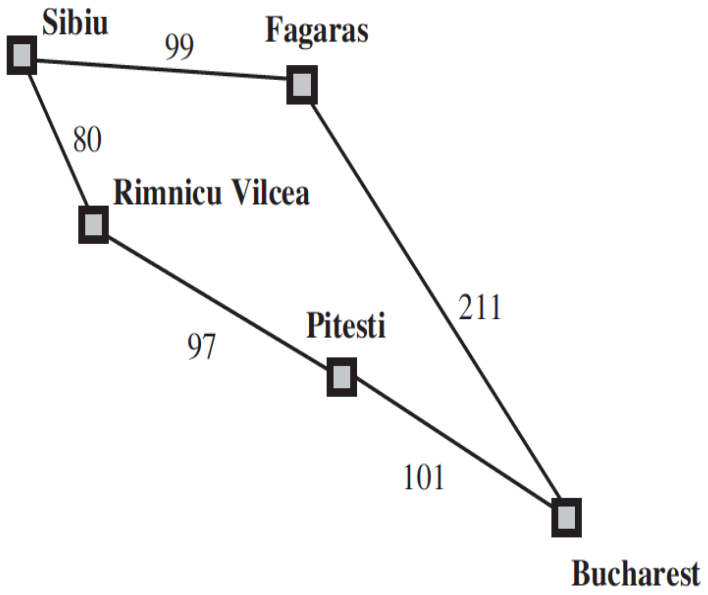
Uniform Cost Search: Cheapest First



- Initial State: Sibiu
- Goal State: Bucharest

- Current State: Fagaras (not a goal state)
- Frontier: [(“Pitesti”, 177)
  (“Bucharest”, 310)]
- Step: Expand “Pitesti” (least cost)
- Result:
  - Generates (“Bucharest”, 278)
  - Replace in Frontier
  - (It’s a Goal State but we won’t test during generation)

Uniform Cost Search: Cheapest First



- Current State: Pitesti (not a goal state)
- Frontier: [(“Bucharest”, 278)]
- Step: Expand “Bucharest”
- Result: No further generation as Goal Test satisfied Return Solution.

- Initial State: Sibiu
- Goal State: Bucharest

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Iterative deepening search

Run Depth Limited Search (DLS) by gradually increasing the limit l

- First with l=1, then l=2, l=3 and so on – until goal is found

Its is a combination of Depth First Search + Breadth First Search

Like DFS, memory requirement is a modest $\mathcal{O}(bd)$ where d is the depth of shallowest goal

Like BFS

- Complete when branching factor is finite
- Optimal when path cost is non decreasing function of depth

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

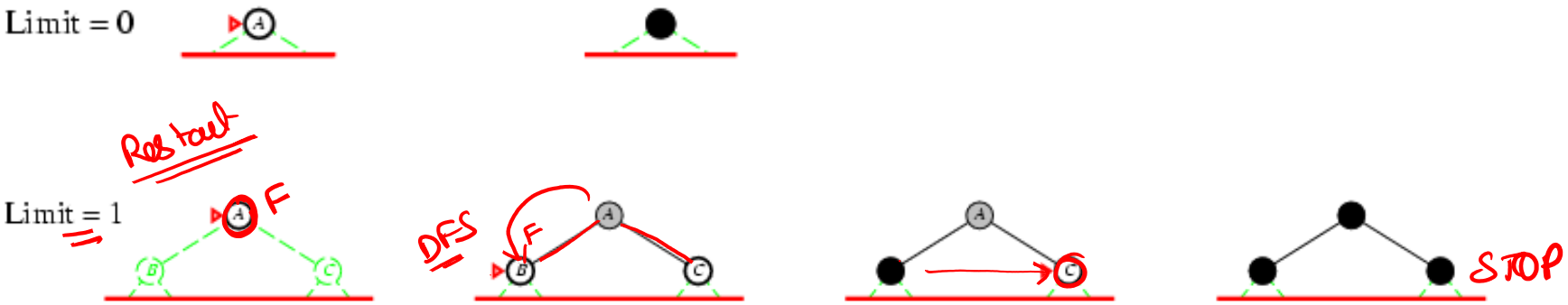
Iterative deepening search

innovate achieve lead

```
function ITERATIVE-DEEPENING-SEARCH( problem) returns a solution, or failure
    inputs: problem, a problem
    for depth ← 0 to ∞ do
        result ← DEPTH-LIMITED-SEARCH( problem, depth)
        if result ≠ cutoff then return result
```

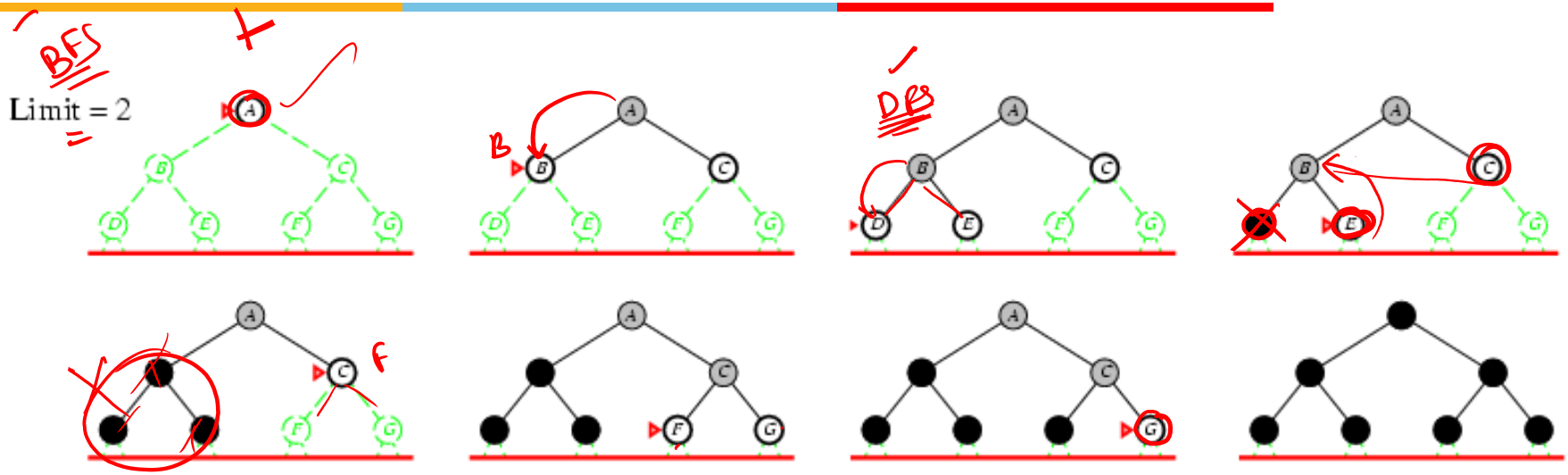
Iterative deepening search l=0

Iterative deepening search l=1



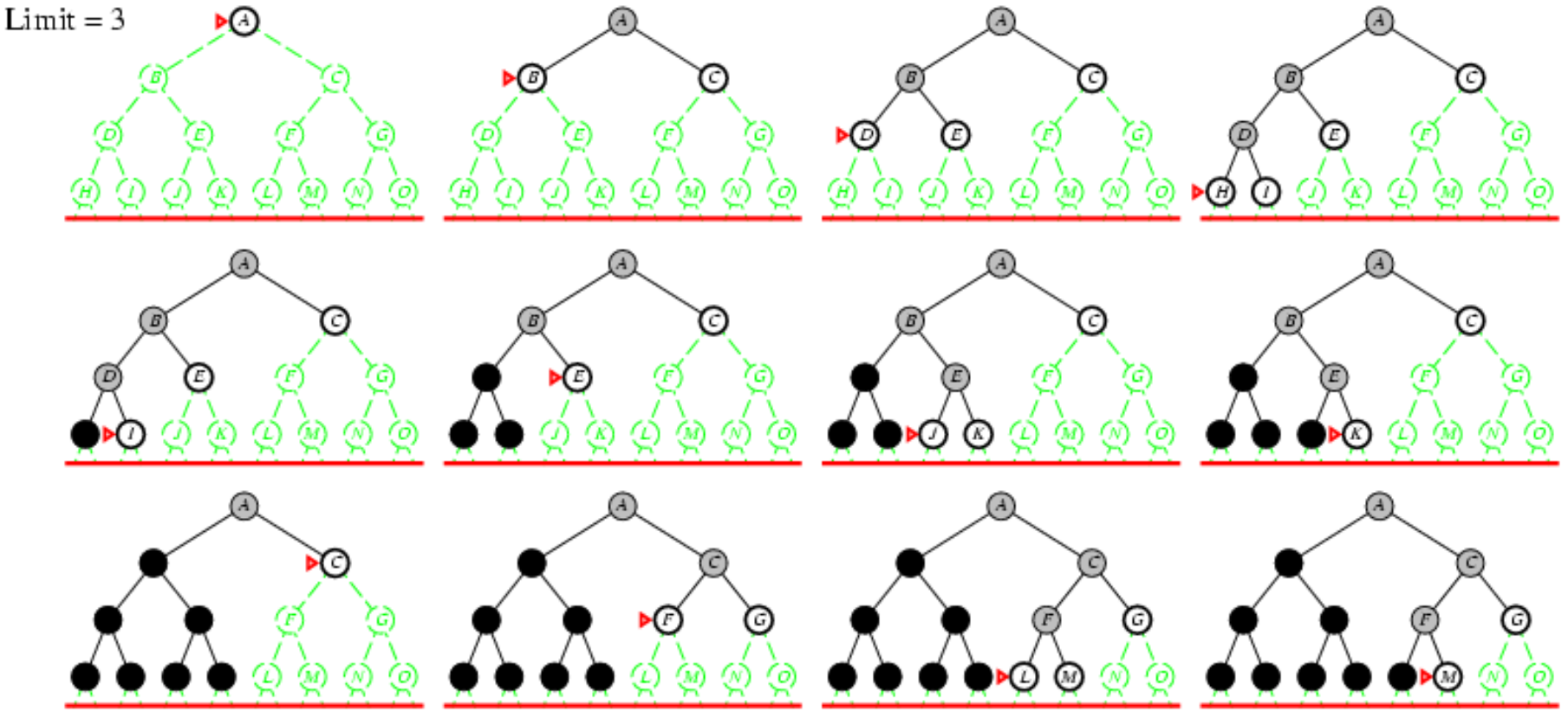
BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Iterative deepening search *l*=2

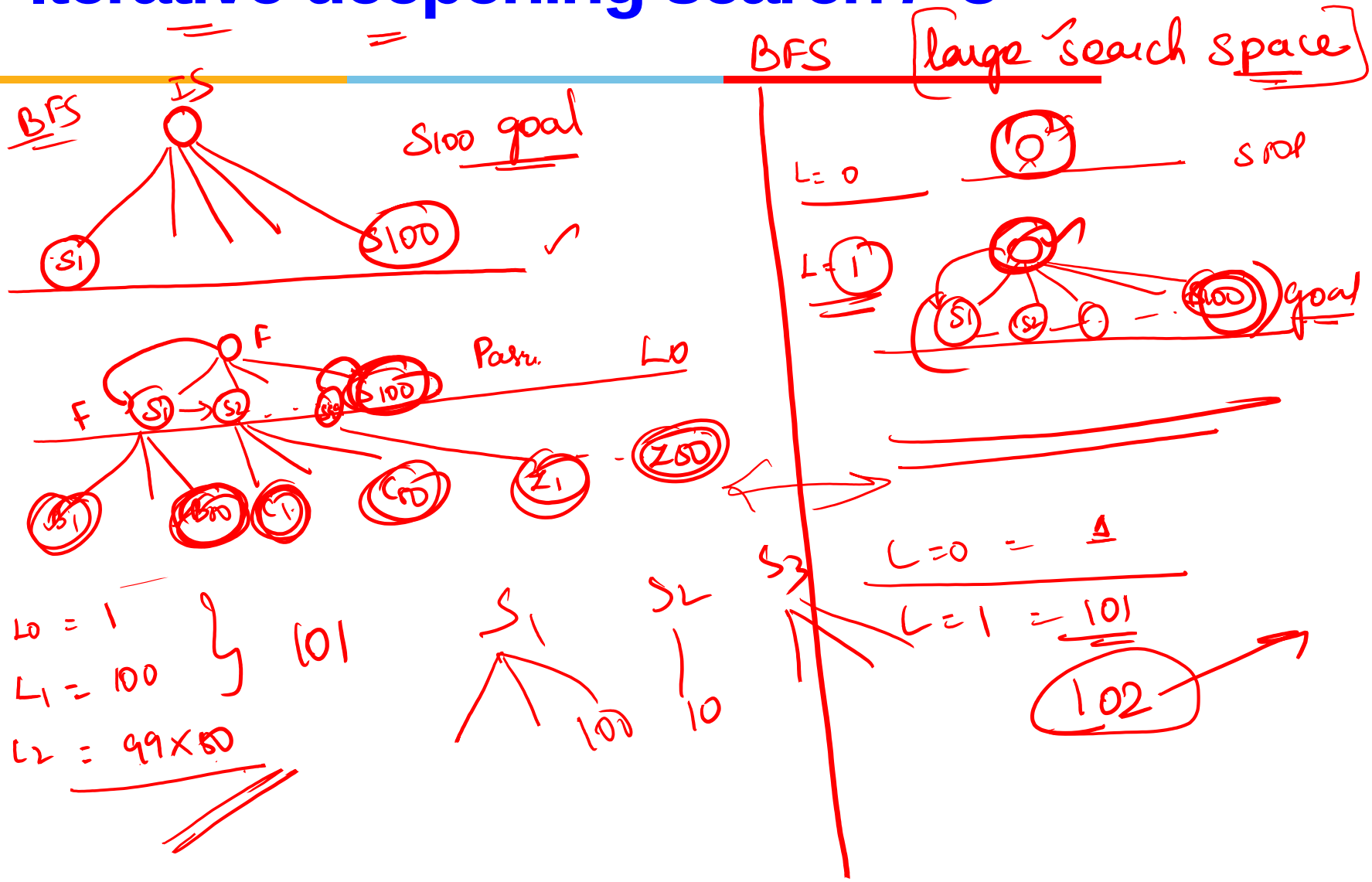


BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Iterative deepening search *l*=3



Iterative deepening search l=3



BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Iterative deepening search

- Complete?
  - Yes
- Time?
  - $-(d+1)b0 + d\,b1 + (d-1)b2 + \ldots + bd = O(b^d)$
- Space?
  - O(bd)

Optimal?

–Yes, if step cost = 1

–Can be modified to explore uniform cost tree (iterative lengthening)

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Uninformed vs Informed Search

- All these methods are slow (blind)

Solution → add guidance ("heuristic estimate")
→ "informed search"

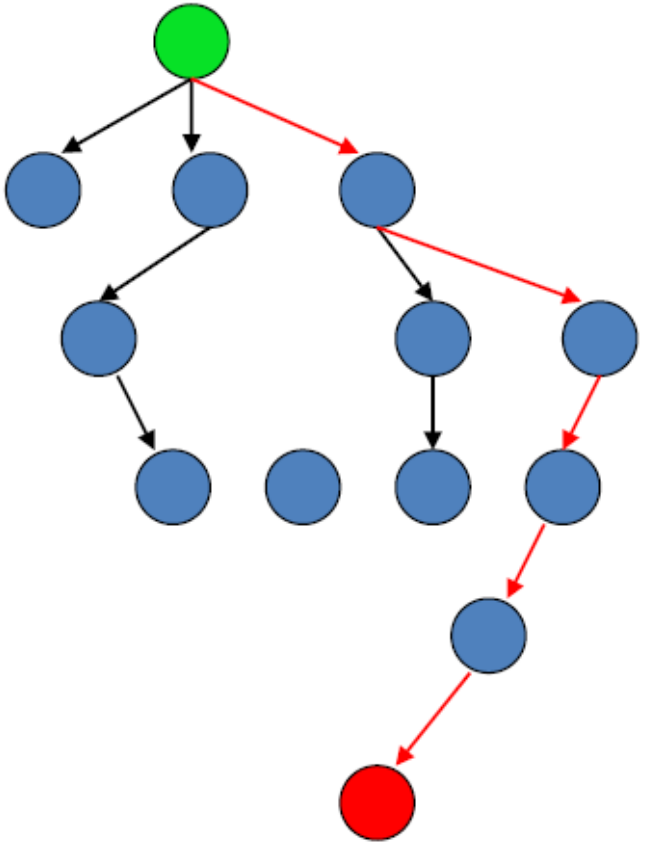
Uninformed vs Informed Search

Parameters	Uninformed Search	Informed Search
Utilizing Knowledge	It does not require using any knowledge during the process of searching.	It uses knowledge during the process of searching.
Speed	Finding the solution is much slower comparatively.	Finding the solution is quicker.
Completion	It is always bound to be complete.	It can be both complete and incomplete.
Consumption of Time	Due to slow searches, it consumes comparatively more time.	Due to a quicker search, it consumes much less time.
Cost Incurred	The expenses are comparatively higher.	The expenses are much lower.
Suggestion/ Direction	The AI does not get any suggestions regarding what solution to find and where to find it	The AI gets suggestions regarding how and where to find a solution to any problem.
Efficiency	It costs more and generates slower results. Thus, it is comparatively less efficient.	It costs less and generates quicker results. Thus, it is comparatively more efficient.
Length of Imp	The implementation is lengthier using AI.	Implementation is shorter using AI.
Examples	DFS, BFS,..	Best First, A*,..

Informed/Heuristic Search

Idea: be **smart** about what paths to try.

A node is selected for expansion based on an evaluation function that estimates cost to goal.

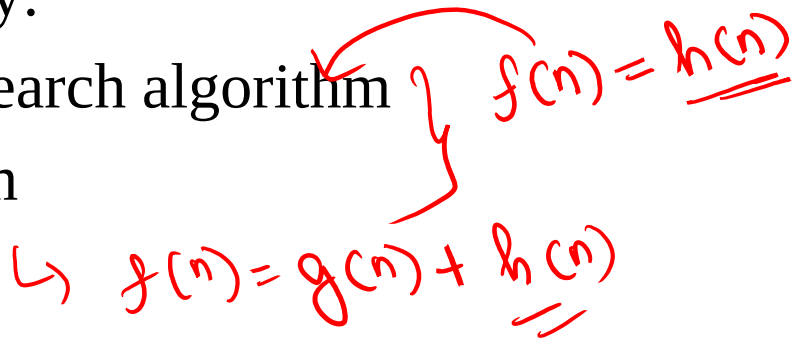


Heuristics Function

- The informed search algorithm is more useful for large search spaces. Informed search algorithm uses the idea of heuristic, so it is also called heuristic search.
- Heuristics Function: Heuristic is a function which is used in informed search, and it finds the most promising path/cost/solution.
- It takes the current state of the agent as its input and produces the estimation of how close the agent is from the goal.
- The heuristic method, however, might not always give the best solution, but it is guaranteed to find a good solution in reasonable time. Heuristic function estimates how close a state is to the goal.

Types of Informed search Algorithms

- There are plenty of them! We will discuss two main informed search algorithms today:
  - Greedy Best First search algorithm
  - A* Search algorithm

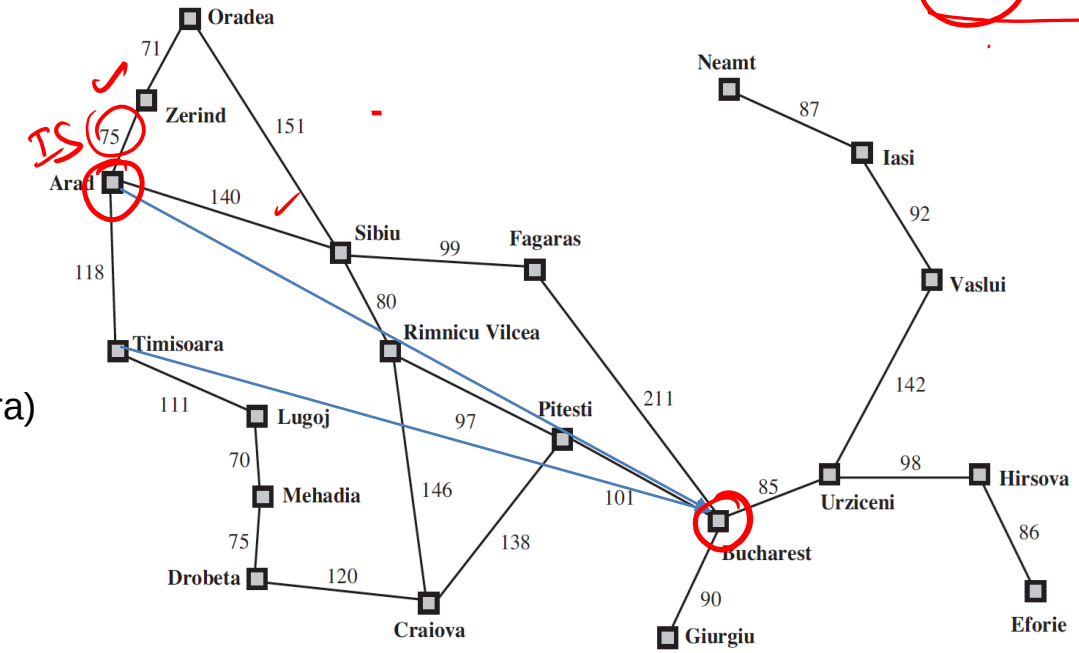


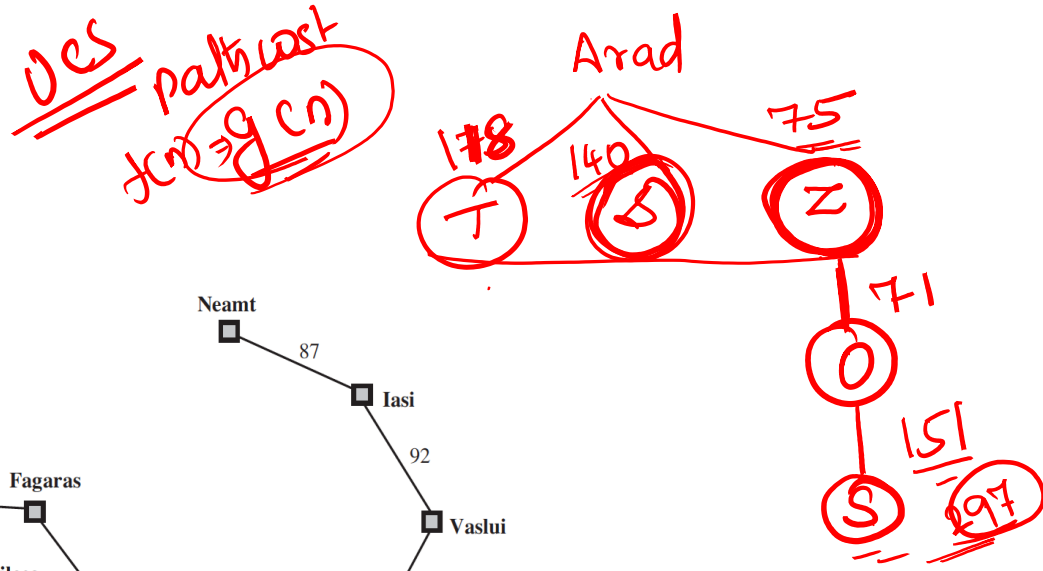
Informed /Heuristic Search

Strategies that know if one non-goal state is more promising than another non-goal state

h(Arad) = 366

h(Timisoara) = 329

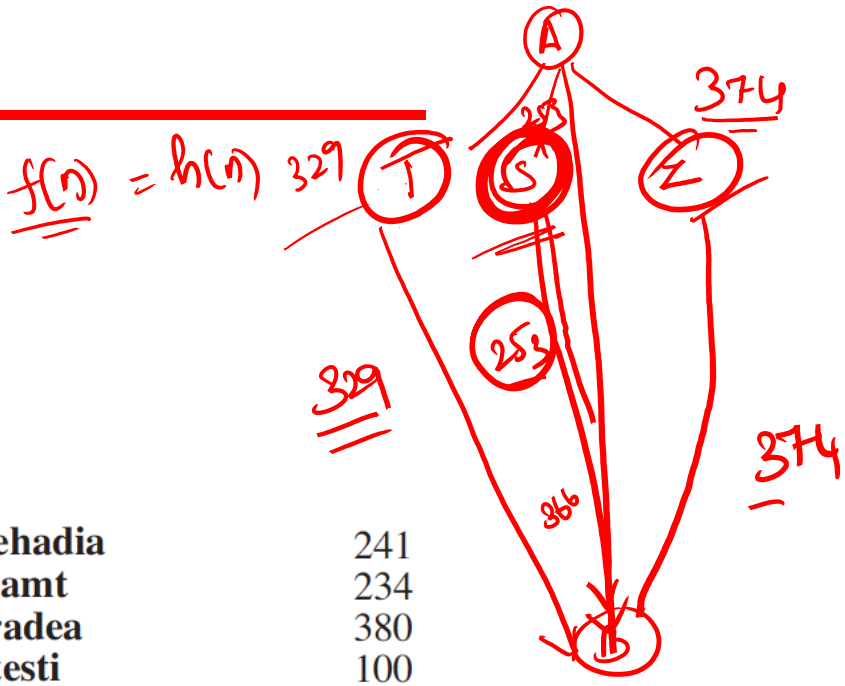




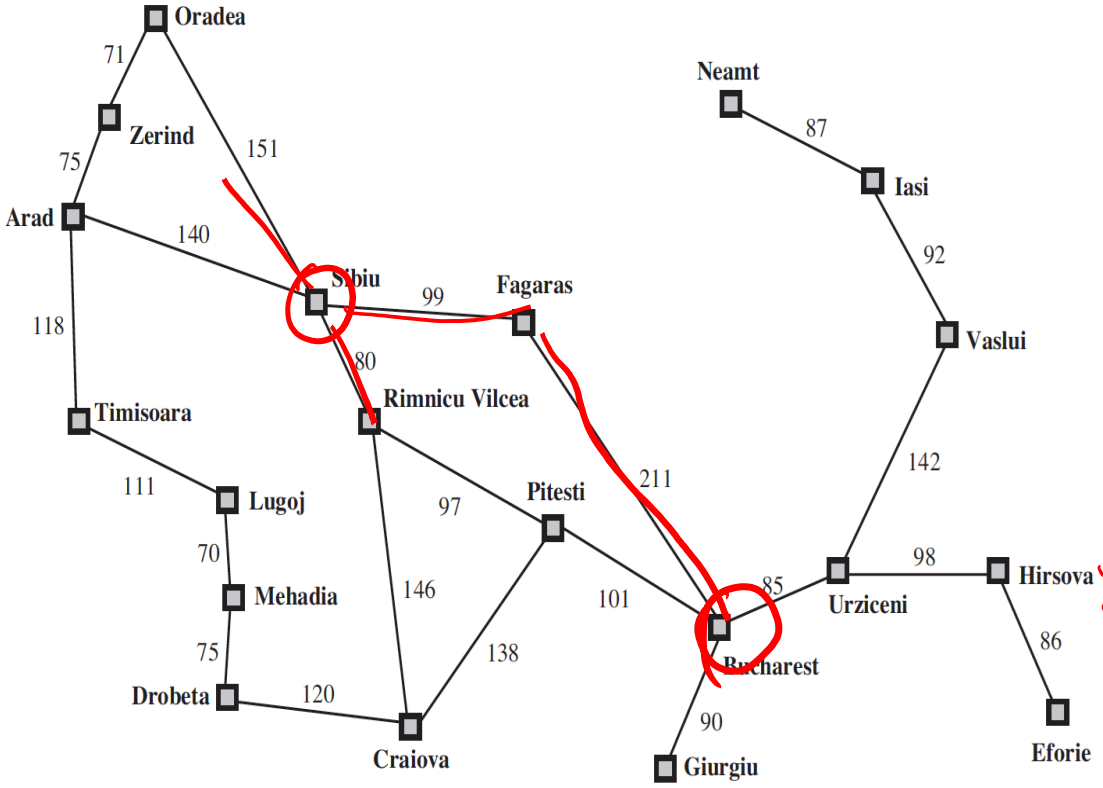
Greedy Best First Search

Expands the node that is closest to the goal

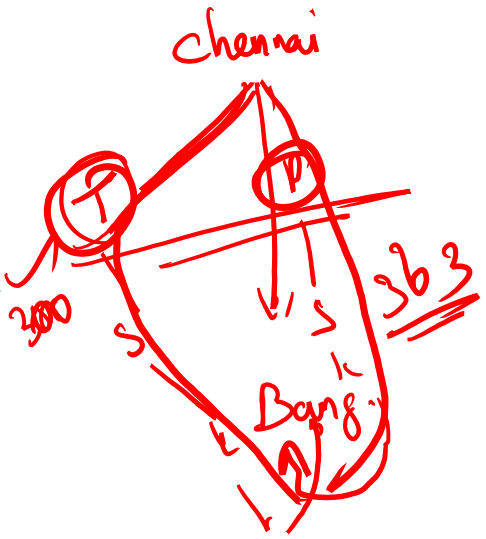
Thus, f(n) = h(n)



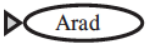
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



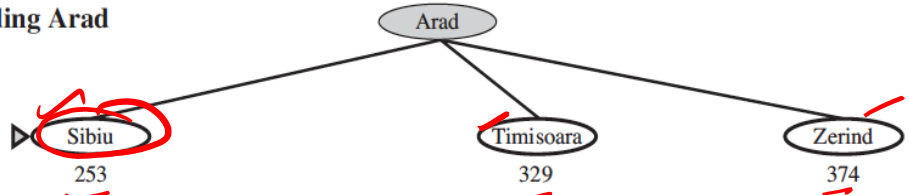
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



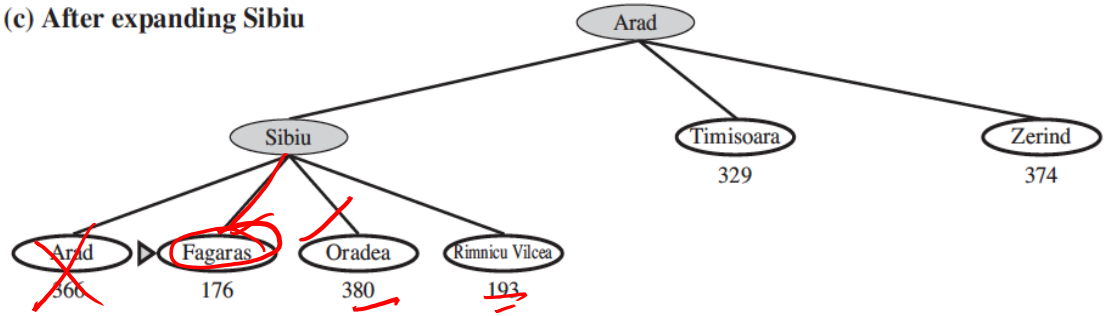
(a) The initial state



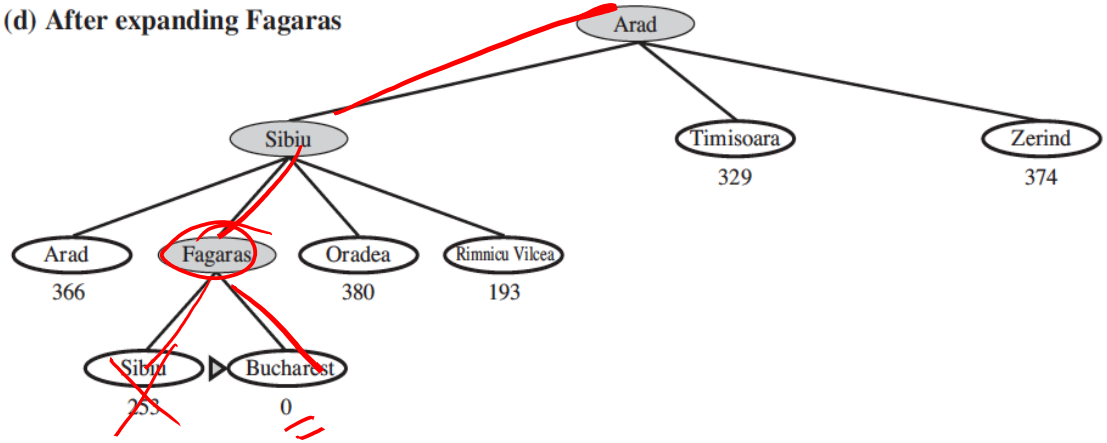
(b) After expanding Arad



(c) After expanding Sibiu



(d) After expanding Fagaras



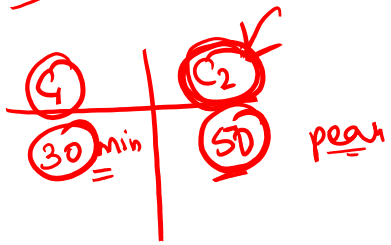
h(n)

GBFS h(n) good

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

A - S - F - B

H - Air



Greedy Best First Search

Not Optimal

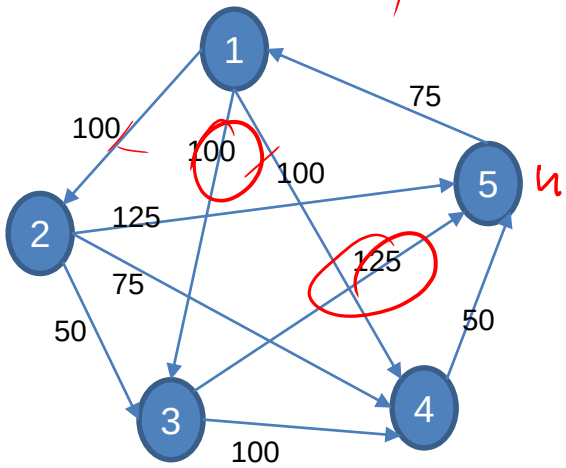
- Because the algorithm is greedy
- It only optimizes for the current action

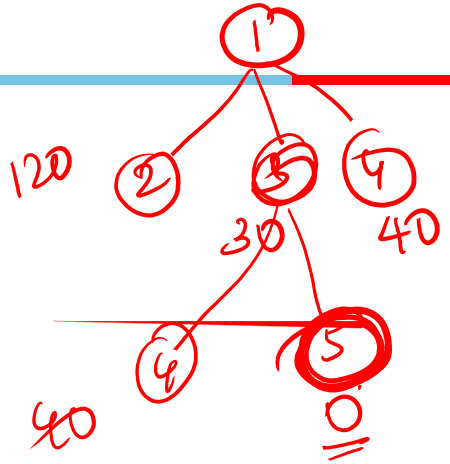
Not Complete

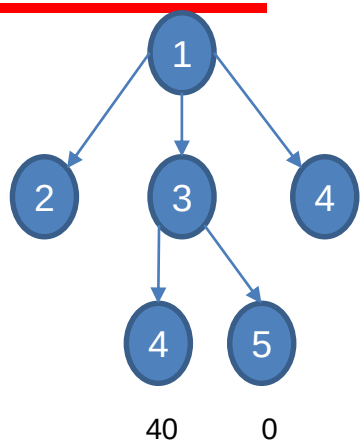
- Often ends up in state with a dead end as the heuristic doesn't guarantee a path but is only an approximation

Time and Space Complexity - $\mathcal{O}(b^m)$ where m – max depth of search tree

Greedy Best First Search







n	h(n)
1	60
2	120
3	30
4	40
5	0

(1)
(1 3) (1 4) (1 2)
(1 3 5) (1 3 4)

C(1-3-5) = 100 + 125 =225
Expanded : 2
Generated : 6
Max Queue Length : 3
Idea: Optimize DFS. Choose next nearest to goal in the same hill.

A* Algorithm

- A* algorithm is one of the best and popular techniques used for path finding and graph traversals.
- A lot of games and web-based maps use this algorithm for finding the shortest path efficiently.
- It is essentially a best first search algorithm.
- This is informed search technique works using heuristic values.
- A* Algorithm works as:
  - It maintains a tree of paths originating as the start node.
  - It extends those paths one edge at a time.
  - It continues until its termination criterion is satisfied.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

A* Algorithm

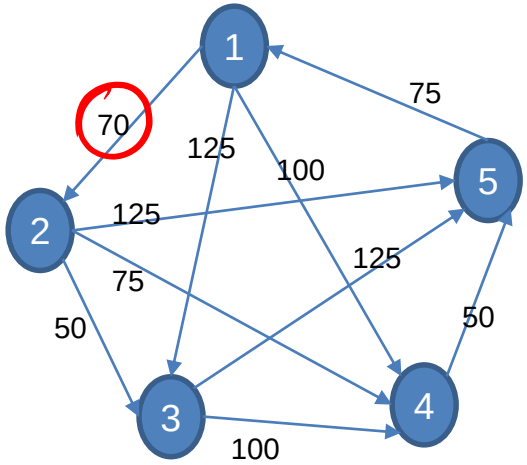
- A* algorithm works on the idea to avoid expanding paths that are already expensive!
- A* algorithm extends the path that minimizes the following function called the evaluation function

$$f(n) = g(n) + h(n)$$

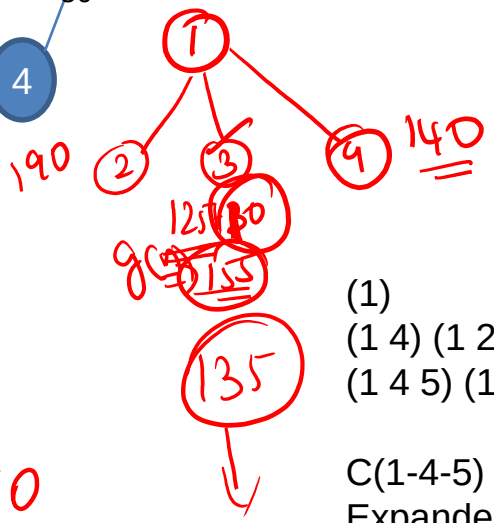
Here,

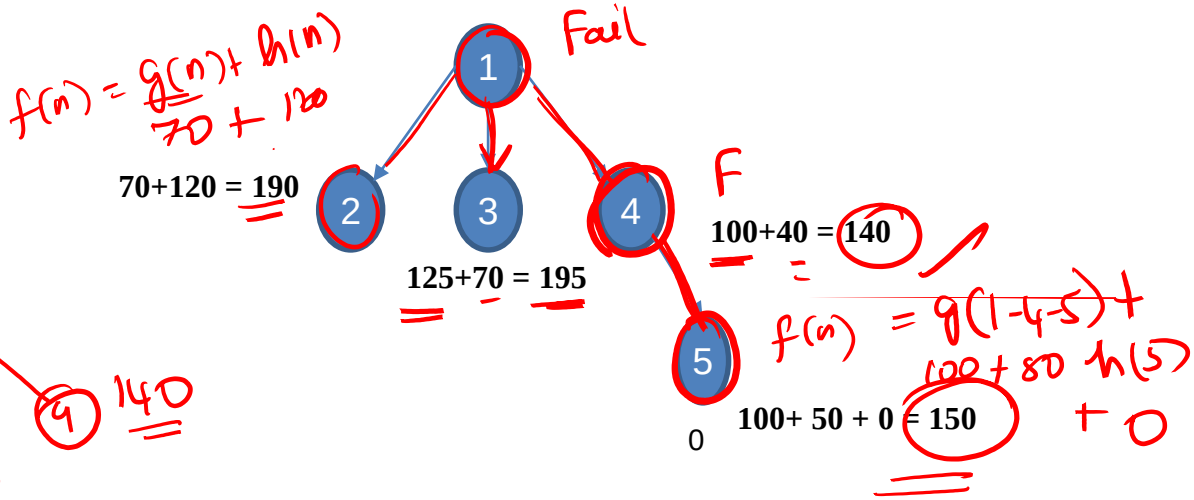
- g(n) is the cost so far to reach n.
- h(n) is the heuristic value i.e. the estimated cost from n to goal.
- f(n) is the estimated total cost of path through n to goal.

A* Search



n	h(n)
1	60
2	120
3	70
4	40
5	0

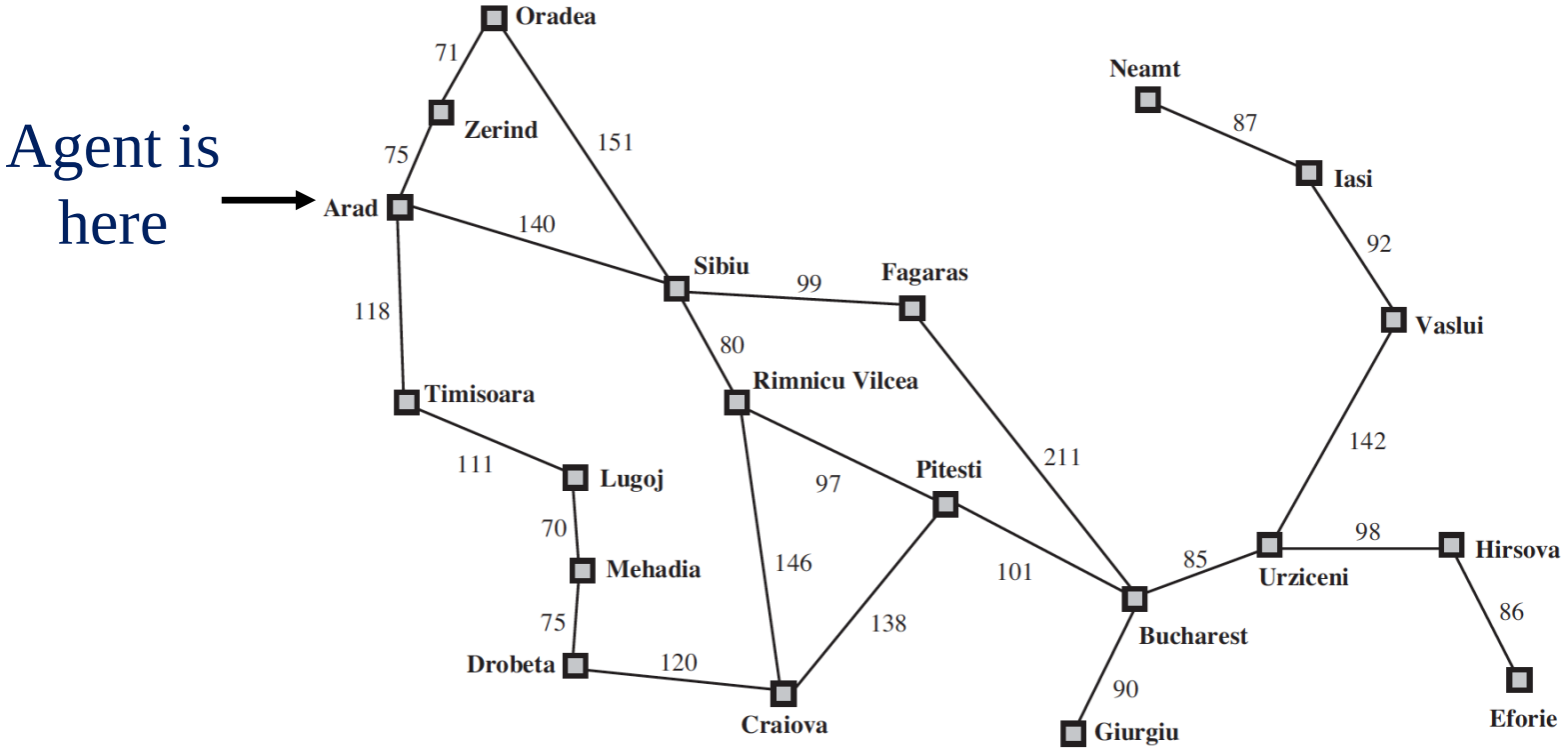




(1)
(1 4) (1 2) (1 3)
(1 4 5) (1 2) (1 3)

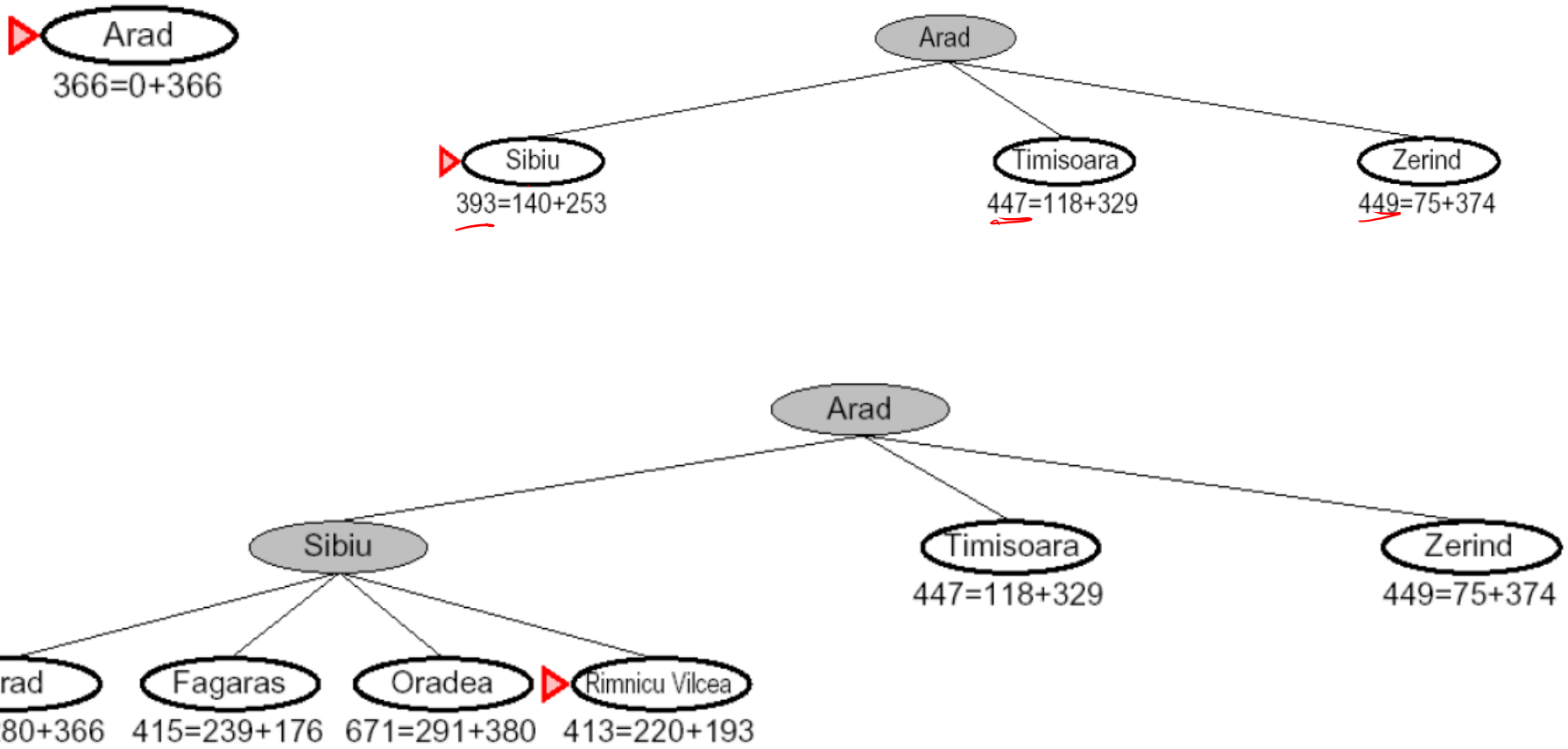
C(1-4-5) = 100 + 150 = 150
Expanded : 2
Generated : 5
Max Queue Length : 3

Example 2 on A* Algorithm



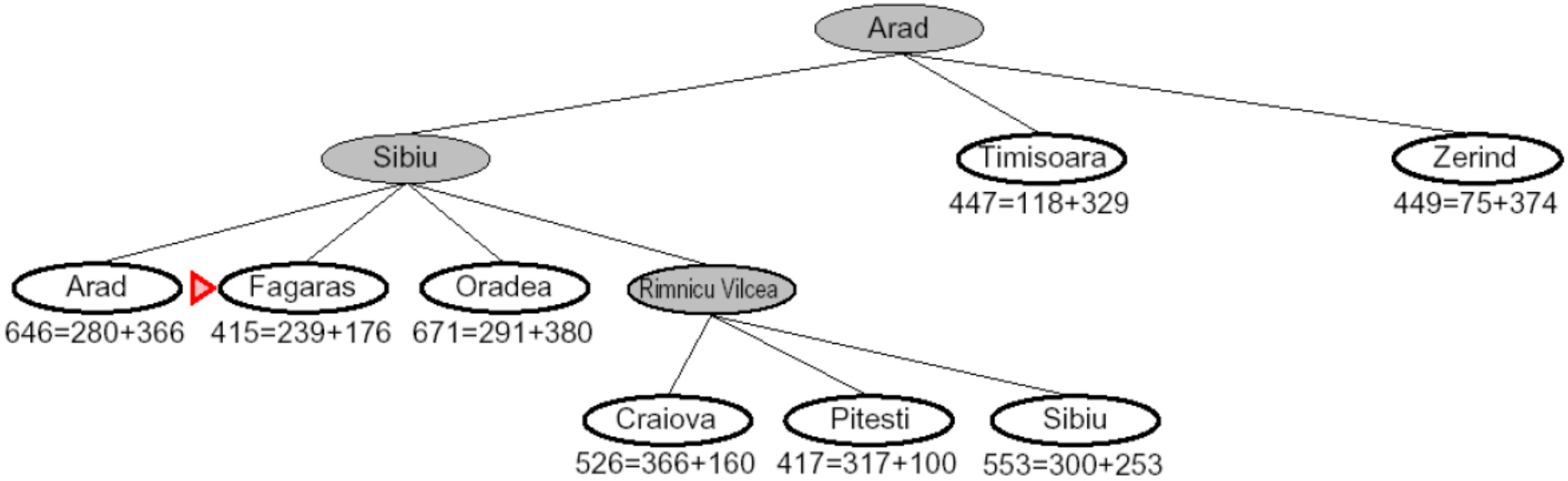
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Example 2: A*



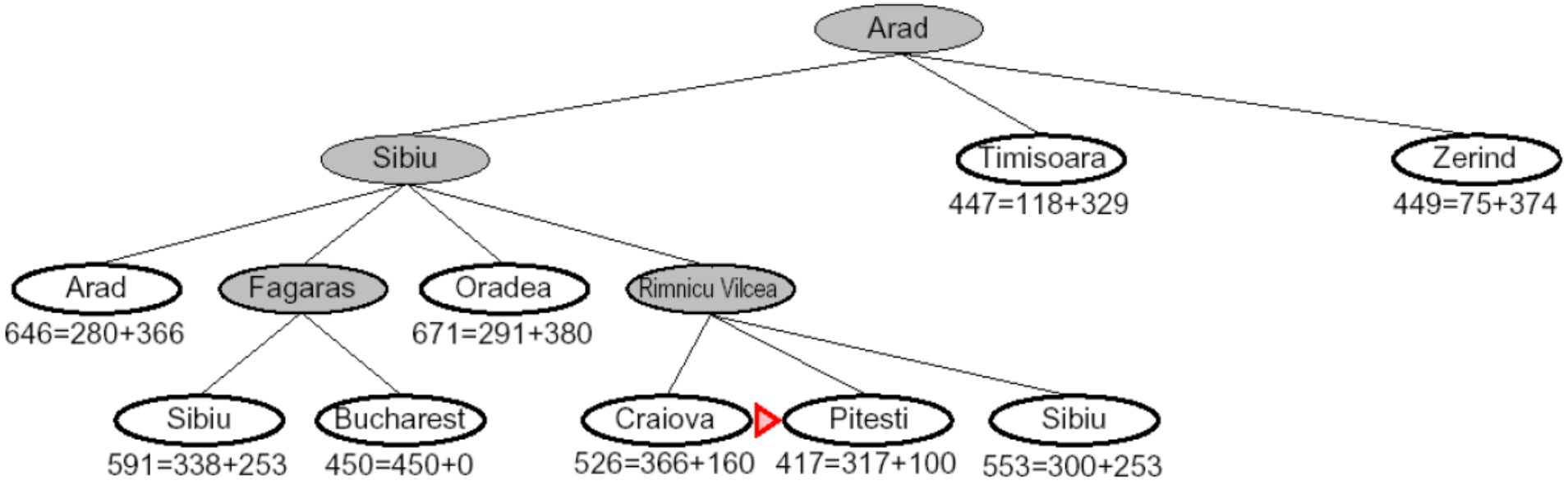
BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Example 2: A*



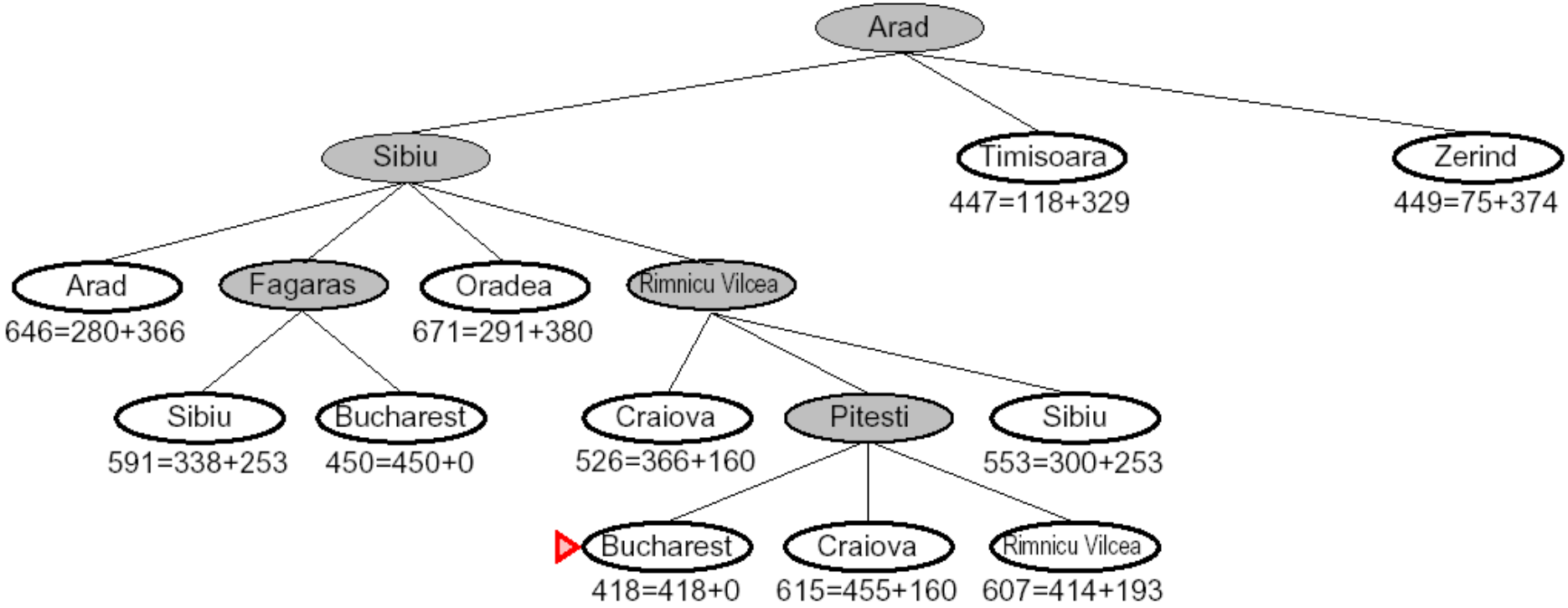
BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Example 2: A*



BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Example 2: A*



BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

When is A* Optimal ?

A* is optimal when the heuristic satisfies certain conditions:

1. **Admissibility**: A heuristic is admissible if it never overestimates the actual cost to reach the goal.

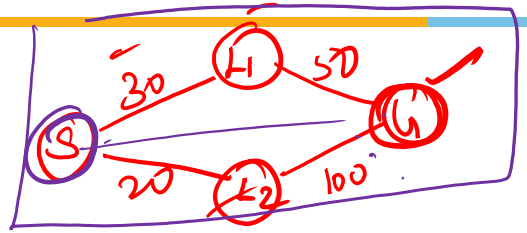
$$h(n) \leq h^*(n),$$

where h*(n) is the true cost from node n to the goal

- **Meaning:** The heuristic is optimistic. It may underestimate the cost, but it should never exaggerate it.
- **Example:** Straight-line distance is admissible for route planning because the actual road distance cannot be shorter than the straight-line distance.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

When is A* Optimal ?



actual env

S_to_G — S-L1-G ⇒ 80 ; S-L2-G ⇒ 120 } 80

actual optimal cost $h^*(n)$

estimated cost $h(n)$

	$h(n)$	$\hat{h}(n)$
S	150	80 ⇒ overestimation
L1	35	50 ⇒ Underestimation
L2	150	100 ⇒ over

When is A* Optimal ?

A* is optimal when the heuristic satisfies certain conditions:

2. **Consistency, also called Monotonicity**: A heuristic is consistent if, for every node n and successor n':

$$h(n) \leq c(\mathrm{n}, \mathrm{n}') + h(n')$$

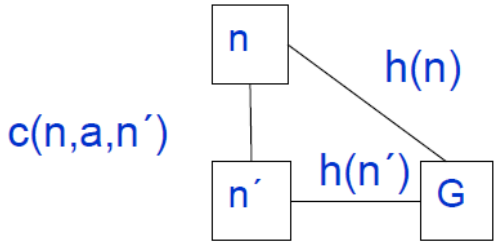
where c(n,n') is the cost of moving from n to n.

- **Meaning:** The estimated cost should not suddenly drop by more than the actual step cost.
- Why Consistency matters: If a heuristic is consistent, f(n) values do not decrease along a path. $\mathrm{f}(n') \geq \mathrm{f}(n)$.
- This ensures that once A* expands a node, the best path to that node has already been found.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Admissible & Consistent Heuristics

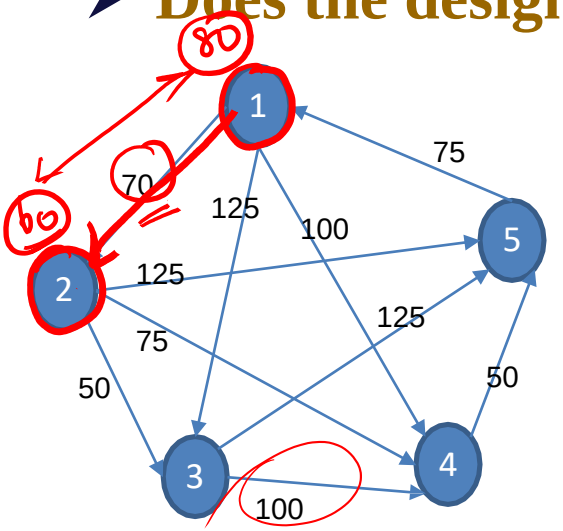
- A heuristic is admissible or optimistic if , **$0 \leq h(n) \leq h^*(n)$**, where h*(n) is the actual cost to reach the goal.
- An admissible heuristic never overestimates the cost to reach the goal.
- h(n) is consistent if
  - for every node n
  - for every successor n´ due to legal action
  - h(n) <= c(n,a,n´) + h(n´)



- Every consistent heuristic is admissible, provided the heuristic value at the goal is zero. However, every admissible heuristic need not be consistent.

A* Search

- **Does the designed heuristic lead to an optimal solution?**



Assuming **node 3** as goal, taking only sample edges per node below is checked for consistency

$g(1,2) = 70$

$h(1) = 80$

$h(2) = 60$

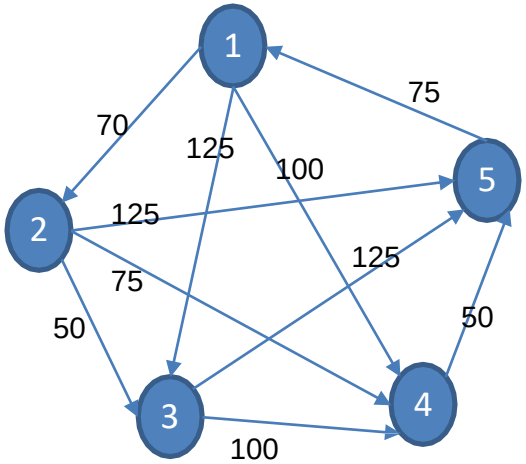
$h(1) - h(2) \leq g(1,2)$

$\Downarrow$

$h(1) \leq g(1,2) + h(2)$

A* Search

- **Does the designed heuristic lead to an optimal solution?**

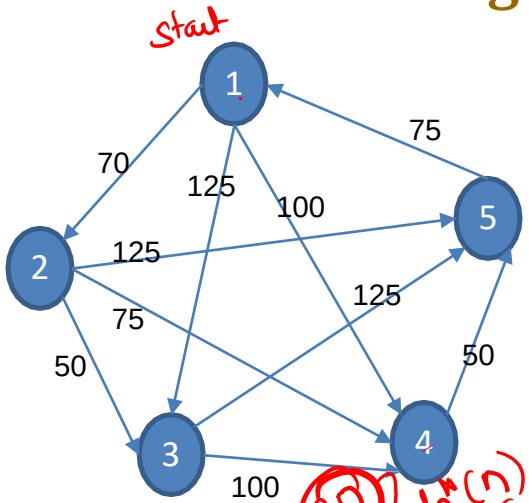


Assuming **node 3** as goal, taking only sample edges per node below is checked for consistency

n	h(n)	Is Admissible? h(n) <= h*(n)	Is Consistent? For every arc (i,j): h(i) <= g(i,j) + h(j)
1			
2			
3			
4			
5			

A* Search

- **Does the designed heuristic lead to an optimal solution?**

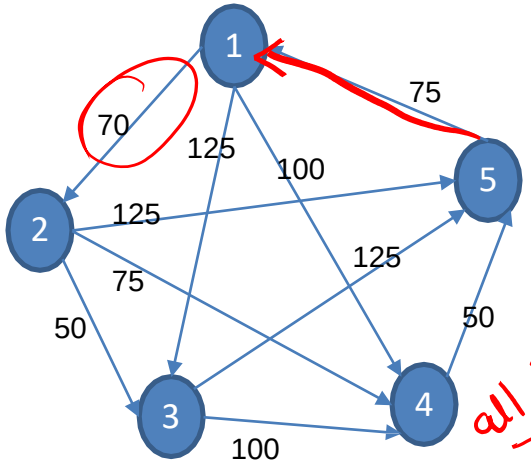


Assuming **node 3** as goal, taking only sample edges per node below is checked for consistency

n	h(n)	Is Admissible? h(n) <= h*(n)	Is Consistent? For every arc (i,j): h(i) <= g(i,j) + h(j)
1	80	Y	N (5→1) : 190<=155
2	60	N	Y (1->2) : 80<=130
3	0	Y	
4	200	Y	Y (1->4) : 80<=300 Y (2->4) : 60<=275
5	190	Y	Y (2->5) : 60<=315 Y (4->5) : 200<=240

A* Search

- **Does the designed heuristic lead to an optimal solution?**



Assuming **node 3** as goal, taking only sample edges per node below is checked for consistency

n	h(n)	Is Admissible? h(n) <= h*(n)	Is Consistent? For every arc (i,j): h(i) <= g(i,j) + h(j)
1	80	Y	N (5→1) : 190<=155
2	60	N	Y (1->2) : 80<=130
3	0	Y	
4	200	Y	Y (1->4) : 80<=300 Y (2->4) : 60<=275
5	190	Y	Y (2->5) : 60<=315 Y (4->5) : 200<=240

What Makes a Good Heuristic?

- A good heuristic helps A* reach the goal faster while still preserving correctness. A good heuristic should be:
  1. **Admissible:** It should never overestimate the true cost.
  2. **Consistent:** It should satisfy the triangle inequality across neighbouring nodes.
  3. **Informative:** It should be close to the actual cost.
- A heuristic that gives values closer to the true cost helps A* avoid unnecessary exploration.

$$h_2(n) \geq h_1(n)$$

- If both heuristics are admissible, then h_2 is usually better because it gives a stronger estimate.

Example:

- For route planning: $h_1(n) = 0$, This becomes Uniform Cost Search.
- $h_2(n)$ = straight-line distance to goal, This gives better guidance and explores fewer nodes.

Good vs Poor Heuristics

Heuristic Type	Effect on A*
h(n) = 0	Always optimal, but slow. Same as Uniform Cost Search.
Admissible but weak heuristic	Optimal, but may explore many nodes.
Admissible and informative heuristic	Optimal and efficient.
Overestimating heuristic	May be faster, but can lose optimality.
Inconsistent heuristic	May require reopening nodes to preserve optimality.

Good heuristic intuition

A good heuristic is like a helpful guide. It does not lie about the distance, but it gives enough direction to avoid exploring everywhere.

Exercise 1

For the below tasks:

1. Planning a trip with a delivery drone
2. Solving a maze
3. Scheduling patient appointments in a clinic

Formulae **the problem** by specifying:

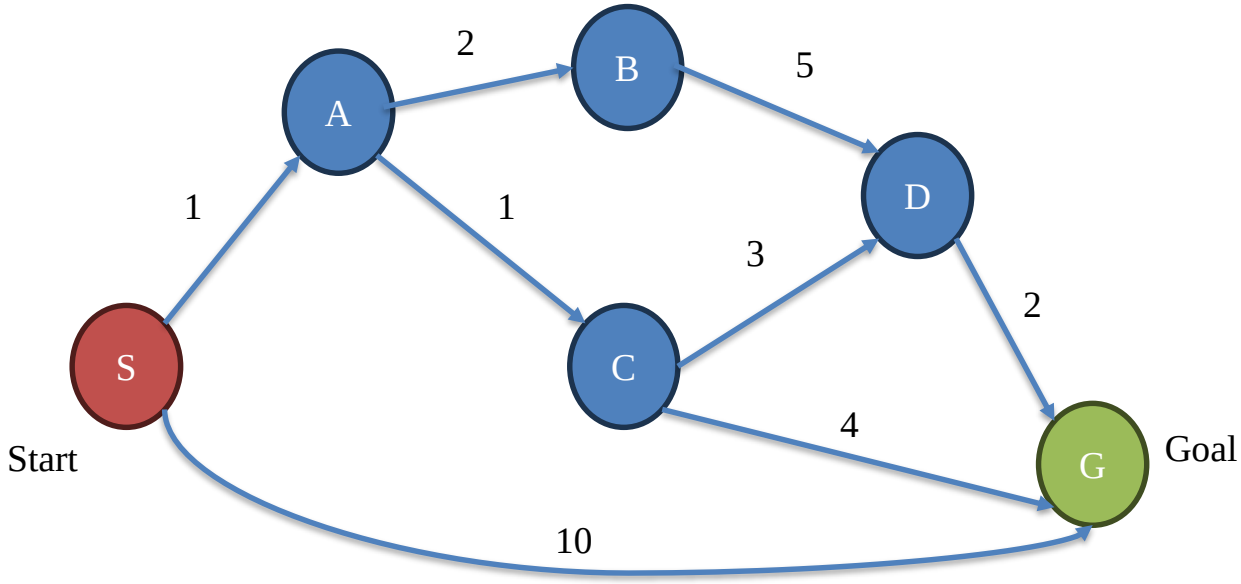
- Initial state
- Possible actions
- Transition model (effects of actions)
- Goal test
- Path cost (optional)

Think about:

- Any **missing components**
- Assess **well-formulated-ness** (Does it meet criteria for a well-formed problem?)

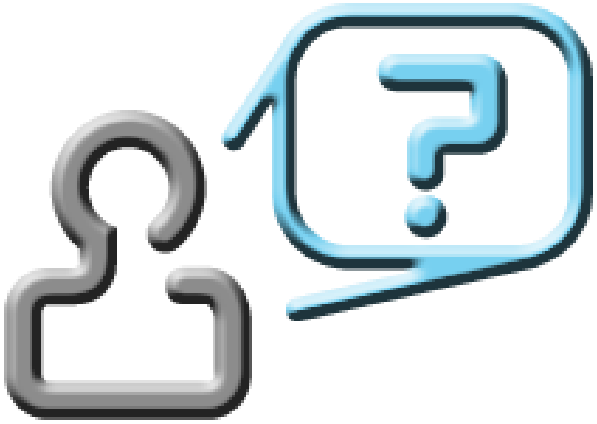
BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Exercise 2



Apply A* algorithm on the above graph, given the heuristics: h(S)=5, h(A)=3, h(B)=4, h(C)=2, h(D)=6 and h(G) = 0

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956



We will explore more on Search techniques in the next class …

Thank You for your time & attention !

Slides are Licensed Under : CC BY-NC-SA 4.0